Application of GARCH frameworks in dynamic volatility modeling and forecasting

Erich Forst

WHU – Otto Beisheim School of Management

Vallendar, Germany

and

Ben Litzinger

WHU – Otto Beisheim School of Management

Vallendar, Germany

1st December 2024

Abstract

This assignment applies logarithmic returns derived from stock closing prices to model percentage changes and facilitate volatility analysis. Tests for ARCH effects confirm the presence of autoregressive conditional heteroscedasticity, consistent with volatility clustering. A GARCH(2,2) model is estimated, with parameter stability evaluated under alternative optimization algorithms to assess the impact of solver choice. This model is also used to forecast the financial time series volatility. A Monte Carlo simulation examines the properties of long-run variance estimates derived from a simulated GARCH(1,1) process across repeated iterations. The analysis demonstrates the practical application of GARCH models for understanding and forecasting conditional volatility in financial markets.

Contents

1 Preparing stock price data for Eaton Corporation (ETN)

1.1 Downloading and visualizing historical stock closing prices

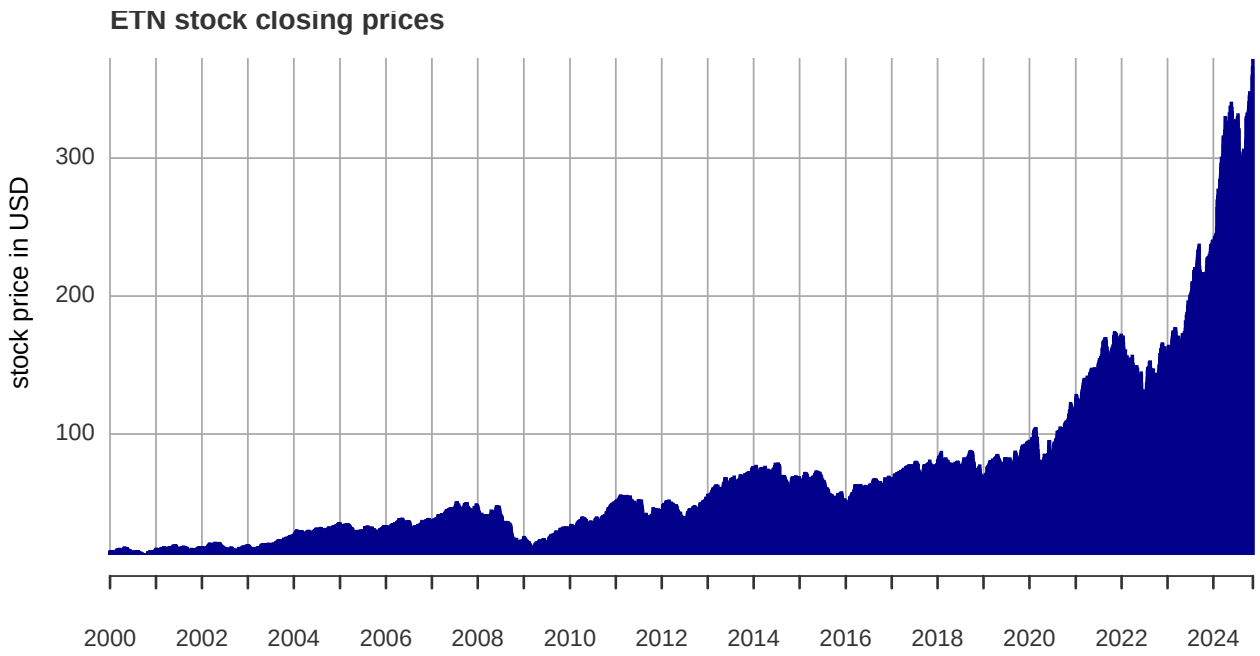
The historical stock price data for Eaton Corporation (ETN) is downloaded from Yahoo Finance for the period January 1, 2000 to November 11, 2024. Non-trading days, such as weekends and holidays, are automatically excluded from the data, eliminating the need for additional filtering. Yahoo Finance data is returned as an "xts" object by default. However, the code includes an additional check to ensure the data is explicitly in "xts" format. The plot of ETN's closing prices provides a visual summary of the stock's historical performance.

```
library(quantmod)
#specify of stock ticker symbol
ticker <- "ETN"
#download of yahoo finance data
getSymbols(ticker,
           from="2000-01-01",
           to="2024-11-12",
           src="yahoo")
```

```
## [1] "ETN"
```

```
#conversion into xts object
if (!inherits(get(ticker), "xts")) {
  assign(ticker, as.xts(get(ticker)))
}
```

```
#plot of the stock price development
plot(ETN$ETN.Close,
     main="ETN stock closing prices",
     ylab="stock price in USD",
     col="darkblue",
     main.timespan=FALSE,
     format.labels="%Y",
     yaxis.right=FALSE,
     type="h")
```



1.2 Transforming stock closing prices into log return series

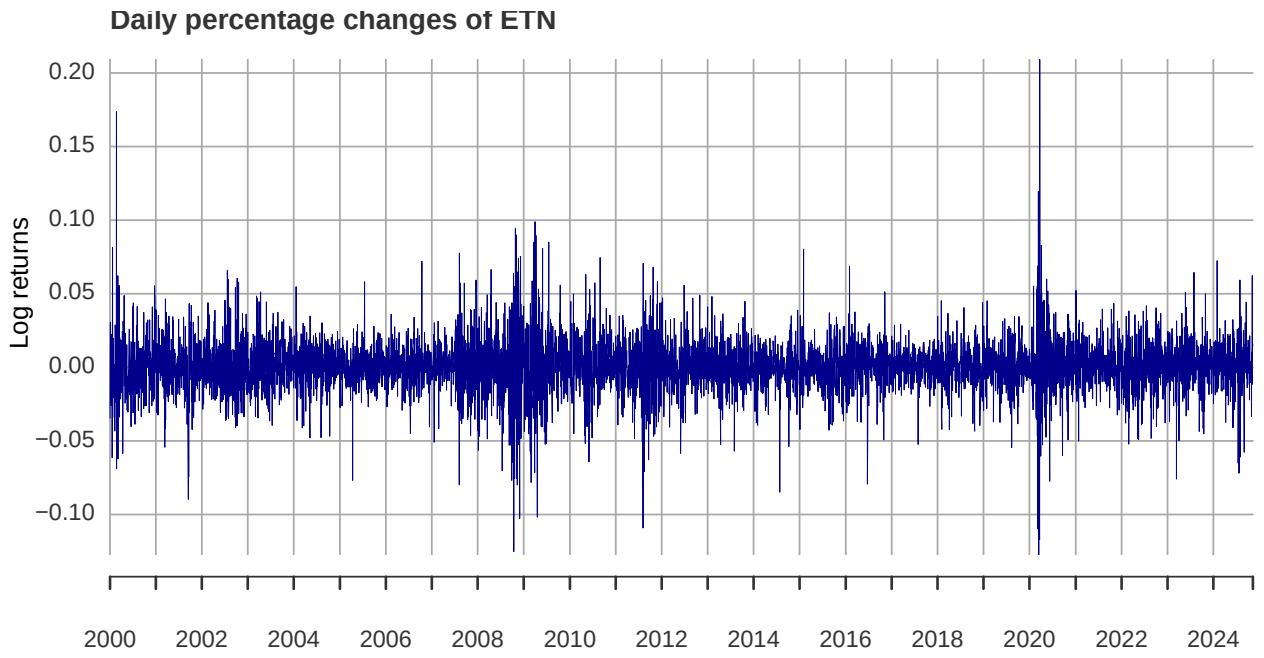
Log returns of Eaton Corporation's stock, denoted as R_t, are computed to approximate the daily percentage changes in its closing prices:

$$R_t = \ln(P_t) - \ln(P_{t-1}) = \ln\left(\frac{P_t}{P_{t-1}}\right) \tag{1}$$

where P_t represents the closing price at time t, and P_{t-1} is the closing price at the previous time period. This transformation expresses the relative change between consecutive closing prices on a logarithmic scale.

The resulting series includes a missing value for the first observation since the calculation requires a lagged price (P_{t-1}) that is unavailable for the initial time step. This missing value is removed to have no NAs in the dataset. By visualizing the log returns, the daily percentage changes in Eaton Corporation's stock prices highlight periods of high and low volatility.

```
#transform closing prices into log returns
ETN$log_returns <- diff.xts(log(ETN$ETN.Close),
                            lag=1,
                            differences=1)
#omit NAs
ETN <- na.omit(ETN)
#plot log returns of closing price
plot(ETN$log_returns,
     main="Daily percentage changes of ETN",
     main.timespan=FALSE,
     ylab="Log returns",
     col="darkblue",
     lwd=0.5,
     format.labels="%Y",
     yaxis.right=FALSE)
```

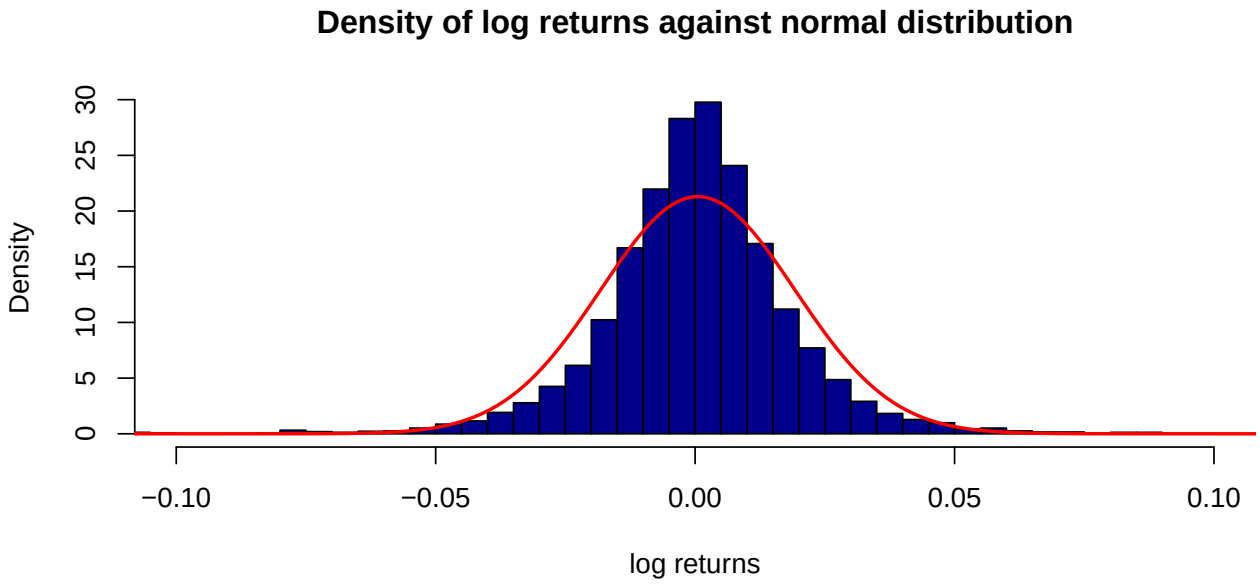


1.3 Analyzing normality in log returns visually

The following analysis visually evaluates whether ETN's log returns approximate a normal distribution. The empirical density is represented by the blue histogram, while the red curve depicts the theoretical normal

density calculated using the sample mean and standard deviation. This comparison provides an intuitive, preliminary check for normality but does not constitute a formal statistical test.

The histogram of ETN's log returns closely resembles a normal distribution; however, the density around the mean is slightly higher than expected under a true normal distribution, indicating a minor deviation from normality.



2 Analyzing autocorrelation and ARCH effects

The analysis evaluates autocorrelation in log returns and squared log returns to identify temporal dependencies and volatility clustering. Autocorrelation quantifies the relationship between current values and past values of a time series at a given lag.

2.1 Assessing temporal dependencies and volatility clustering

For instance, the autocorrelation at lag j is defined as:

$$\rho_j = \frac{\text{Cov}(R_t, R_{t-j})}{\sqrt{\text{Var}(R_t) \cdot \text{Var}(R_{t-j})}} \tag{2}$$

where $\text{Cov}(R_t, R_{t-j})$ is the covariance between log returns at time t and $t-j$, and $\text{Var}(R_t)$ and $\text{Var}(R_{t-j})$ represent their variances.

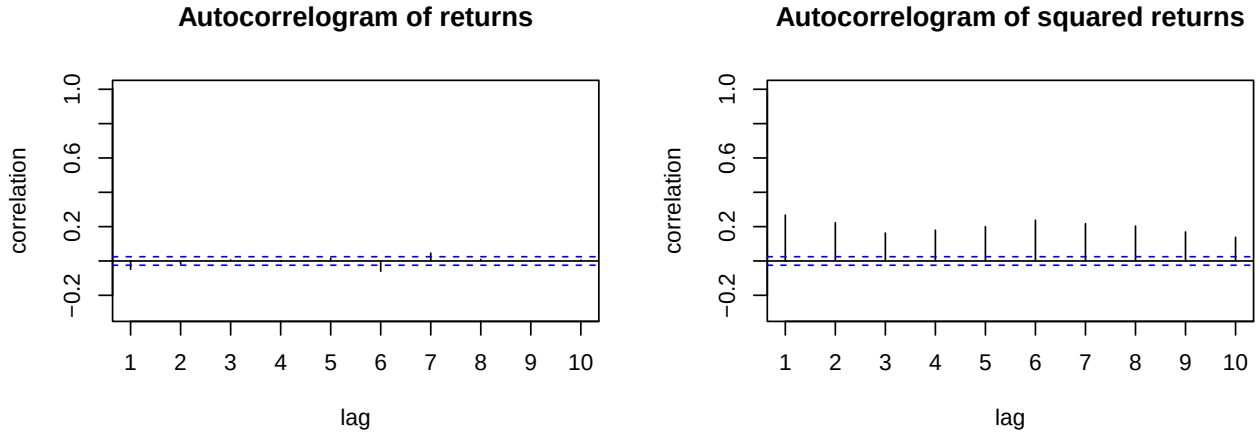
The autocorrelograms reveal distinct patterns: log returns show almost no autocorrelation, indicating they are uncorrelated and unpredictable over time. In contrast, squared log returns exhibit some autocorrelation, reflecting volatility clustering. This phenomenon suggests periods of high or low volatility tend to persist.

```
#use forecast library to create autocorrelograms
library(forecast)

par(mfrow=c(2,2))
```

```r
#plot autocorrelation of log-returns
acf <- Acf(ETN$log_returns,
           lag.max=10,
           type="correlation",
           plot=TRUE,
           main="Autocorrelogram of returns",
           xlab="lag",
           ylab="correlation",
           ylim=c(-0.3, 1))
#square log_returns
ETN$sq_log_returns <- ETN$log_returns**2

#plot autocorrelation of squared log-returns
acf <- Acf(ETN$sq_log_returns,
           lag.max=10,
           type="correlation",
           plot=TRUE,
           main="Autocorrelogram of squared returns",
           xlab="lag",
           ylab="correlation",
           ylim=c(-0.3, 1))
```



2.2 Testing for ARCH effects

To formally test for ARCH effects, the Ljung-Box test is conducted, which evaluates whether the residuals' squared terms (a proxy for variance) are correlated. The null hypothesis (H_0) assumes no autocorrelation in the squared residuals, which implies the absence of ARCH effects. The Box-Ljung test statistic is computed as:

$$LB = n \sum_{i=1}^{m} \frac{n+2}{n-i} r_i^2 \tag{3}$$

where r_i^2 is the squared autocorrelation at lag i, n is the sample size, and m is the number of lags.

This statistic follows a χ^2 distribution with m degrees of freedom.

```r
#perform Box-Ljung to test for ARCH effects
ETN_meanadj <- lm(ETN$log_returns ~ 1)
ETN$meanadj_returns <- ETN_meanadj$residuals
ETN$uhat_sq <- ETN$meanadj_returns**2
Ljung_Box_Test <- Box.test(ETN$uhat_sq,
                           lag=10,
                           type="Ljung-Box")
print(Ljung_Box_Test)
```

```
##
##  Box-Ljung test
##
## data:  ETN$uhat_sq
## X-squared = 2614, df = 10, p-value < 2.2e-16
```

In the analysis, the Box-Ljung test applied to squared residuals yields a p-value of 2.2×10^{-16}, which is substantially below common significance thresholds (e.g., 0.05). Therefore, H_0 is rejected, confirming that past residuals are significantly correlated with the current variance. This result aligns with the visual evidence from the autocorrelograms of squared returns, confirming the presence of volatility clustering and ARCH effects.

2.3 Testing for ARCH effects with the BIC

The following analysis evaluates the Bayesian Information Criterion (BIC) for autoregressive (AR) models of different lag orders, using the log returns of the financial time series data. By decomposing the BIC calculation into its constituent terms, we gain insights into the trade-off between model fit (via residual sum of squares) and complexity (penalizing additional parameters). The results reveal that the lowest BIC value occurs at lag 1, indicating that the AR(1) model offers the best balance between simplicity and explanatory power. Additionally, R^2 values were computed to assess the proportion of variance explained by each model.

Table 1: BIC Values for Different Lags

Lag	term1	term2	BIC	R2
1	-7.95733	0.00280	-7.95453	0.00231
2	-7.95775	0.00419	-7.95356	0.00273
3	-7.95777	0.00559	-7.95218	0.00275
4	-7.95777	0.00699	-7.95078	0.00275
5	-7.95801	0.00839	-7.94962	0.00299
6	-7.96150	0.00978	-7.95172	0.00647
7	-7.96320	0.01118	-7.95201	0.00815
8	-7.96330	0.01258	-7.95073	0.00825
9	-7.96331	0.01398	-7.94934	0.00826
10	-7.96334	0.01537	-7.94796	0.00829

3 Specification and Estimation of a GARCH(2,2) Model

3.1 Comparing optimisation routines for GARCH estimation

To estimate a GARCH(2,2) model, the rugarch library is employed, which facilitates the specification and estimation. The model is designed to capture conditional heteroscedasticity by modeling the conditional variance (σ_t^2) as a function of two lagged squared residuals ($\epsilon_{t-1}^2, \epsilon_{t-2}^2$) and two lagged conditional variances ($\sigma_{t-1}^2, \sigma_{t-2}^2$).

The mean model assumes no autoregressive or moving average components (ARMA(0,0)) and includes only a constant term. This is expressed as:

$$R_t = \mu + \epsilon_t, \tag{4}$$

where R_t is the observed time series (e.g., log returns), μ is the constant mean, and ϵ_t is the error term, which follows a conditional variance structure.

The GARCH(2,2) model is defined by the following equation for the conditional variance:

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \alpha_2 \epsilon_{t-2}^2 + \beta_1 \sigma_{t-1}^2 + \beta_2 \sigma_{t-2}^2, \tag{5}$$

where σ_t^2 is the conditional variance at time t, ω is the constant term, α_1 and α_2 are the coefficients for lagged squared residuals, and β_1 and β_2 are the coefficients for lagged conditional variances.

```
#load library rugarch to use GARCH models
library(rugarch)
#specify garch(2,2)
garch22_spec <- ugarchspec(variance.model=list(garchOrder=c(2,2)),
                           mean.model=list(armaOrder=c(0,0),include.mean=TRUE),
                           distribution.model="norm")
#estimate the model with solnp
garch22_fit_solnp <- ugarchfit(garch22_spec,
                               ETN$log_returns,
                               solver="solnp")
# round(garch22_fit_solnp@fit[["matcoef"]],
#        digits=8)
#estimate the model with lbfgs
garch22_fit_lbfgs <- ugarchfit(garch22_spec,
                               ETN$log_returns,
                               solver="lbfgs",
                               solver.control=list(pgtol=0.3, maxit = 10e5))
# round(garch22_fit_lbfgs@fit[["matcoef"]],
#        digits=8)
```

3.2 Interpreting the signifance of the estimators

After estimating the GARCH(2,2) model using both the *solnp* and *lbfgs* optimization routine, the significance of the estimated parameters can be judged. Notably, the coefficient α_1, which represents the impact of past trading day's shocks (or innovations), is highly significant with a p-value close to zero for both optimisation routines. Surprisingly, the second lagged residual term, α_2, is entirely insignificant ($p = 1$), indicating that shocks from two days ago have no measurable effect on the conditional variance.

The parameter β_1 which represents the persistence of volatility from the previous period is found to be statistically significant for *lbfgs* routine with a p-value of 0.001. However, in the *solnp* routine β_1 is not statistically significant with a p-value of 0.514.

Similarly, the coefficient β_2 which captures the effect of volatility from two periods ago, is significant for lbfgs with a p-value of 0.020, whereas for solnp, it is not significant, with a p-value of 0.606.

The estimation of the constant is also significant for *lbfgs (p=0)* but not for *solnp (p=0.179).*

The inconsistent results for estimating β_0 and β_0 may be attributed to the specific solver control settings. The lbfgs routine optimization was performed with the settings *solver.control=list(pgtol=0.3, maxit=10e5)* which allowed for a larger number of iterations and a higher tolerance for convergence.

A larger pgtol value (like 0.3) means the optimization process can stop earlier, even if the gradient isn't close to zero. This can lead to faster convergence, but at the risk of stopping before the true optimal solution is found. 0.3 was the smallest possible value that allowed the optimization to converge.

The value for maxit sets the maximum number of iterations before stopping the optimization process, ensuring the solver has enough opportunities to find the best-fitting parameters.

Table 2: Comparison of GARCH(2,2) Parameter Estimates from 'lbfgs' and 'solnp' Solvers

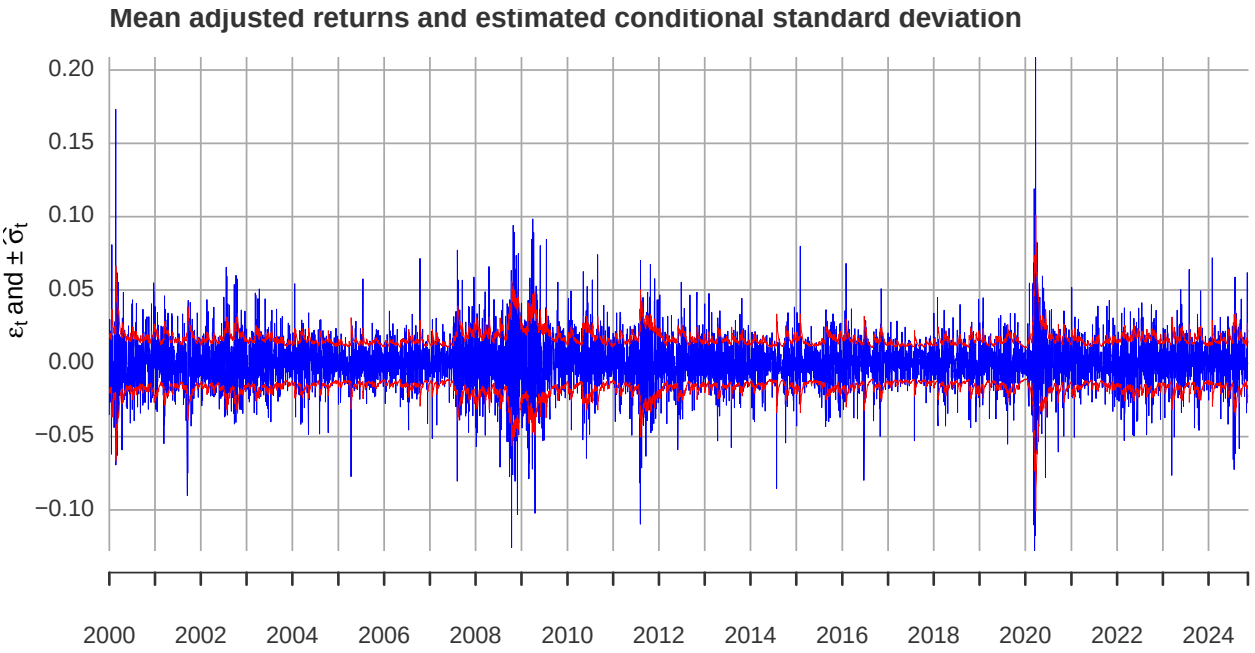
Parameter	lbfgs_estimate	lbfgs_p_value	solnp_estimate	solnp_p_value
omega	0.00002	0	0.00002	0.179
alpha1	0.136	0	0.135	0
alpha2	0	1	0.00000	1
beta1	0.490	0.001	0.491	0.514
beta2	0.319	0.020	0.318	0.606

4 Plotting and forecasting the conditional standard deviation

4.1 Plotting mean adjusted returns with superimposed standard deviations

To analyze the dynamics of time-varying volatility in financial returns, the mean-adjusted returns are plotted alongside the estimated conditional standard deviation $\pm\hat{\sigma}_t$, derived from a GARCH(2,2) model.

```
#extract positive and negative estimated sigma from GARCH fit
ETN$sig_t_hat_garch22 <- sigma(garch22_fit_solnp)
ETN$neg_sig_t_hat_garch22 <- -ETN$sig_t_hat_garch22
#plot mean-adjusted returns with superimposed +/- estimated sigmas
plot(ETN[, c('sig_t_hat_garch22', 'neg_sig_t_hat_garch22', 'meanadj_returns')],
     ylab=expression(epsilon[t] ~ "and" ~ "\u00B1" ~ widehat(sigma)[t]),
     lwd=c(0.5, 0.5, 0.25),
     col=c('red', 'red', 'blue'),
     format.labels="%Y",
     main="Mean adjusted returns and estimated conditional standard deviation",
     main.timespan=FALSE,
     yaxis.right=FALSE,
     )
```



4.2 Forecasting volatility with a GARCH(2,2) model

Next, a one-step-ahead forecast is computed, which provides the forecasted volatility, $\hat{\sigma}_{t+1}$.

The equation for the conditional variance in the GARCH(2,2) model is:

$$\hat{\sigma}^2_{t+1} = \omega + \alpha_1\epsilon^2_t + \alpha_2\epsilon^2_{t-1} + \beta_1\sigma^2_t + \beta_2\sigma^2_{t-1} \tag{6}$$

```
#forecast next day with GARCH(2,2) fit
garch22_fcast <- ugarchforecast(garch22_fit_solnp, n.ahead=1)
fcast_val <- as.numeric(garch22_fcast@forecast$sigmaFor)
#input the forecasted value into the existing matrix
last_date <- tail(index(ETN), 1)
```

```r
next_date <- last_date + 1
ETN$fcast_val <- NA
new_row <- xts(matrix(NA, ncol=ncol(ETN)), order.by=next_date)
colnames(new_row) <- colnames(ETN)
new_row[1, "fcast_val"] <- fcast_val
ETN <- rbind(ETN, new_row)
```

5 Simulating and estimating a GARCH(1,1) model

5.1 Simulating Monte Carlo Observations for Analyzing GARCH Model Properties

In this analysis, a GARCH(1,1) model is estimated and simulated to evaluate the properties of parameter estimates and the unconditional variance.

Using the predefined parameter values $\omega = 0.1$, $\alpha_1 = 0.199999$, and $\beta_1 = 0.8$, 5000 observations are simulated from the GARCH model. The simulated data is fitted onto a GARCH(1,1) model with unknown parameters. The resulting parameter estimates for ω, α_1, and β_1 are extracted.

The unconditional variance is calculated for each simulation run to assess the stationarity condition of the model:

$$\text{Var}(R_t) = \frac{\omega}{1 - \alpha_1 - \beta_1} \tag{7}$$

Intuitively, it captures the balance between the constant term (ω) and the contributions of past shocks (α_1) and past variances (β_1). For the model to be stationary, the condition $\alpha_1 + \beta_1 < 1$ must hold; otherwise, the denominator would approach zero or become negative, leading to an undefined or infinite variance.

This process is repeated 1,000 times, saving the estimations into a matrix.

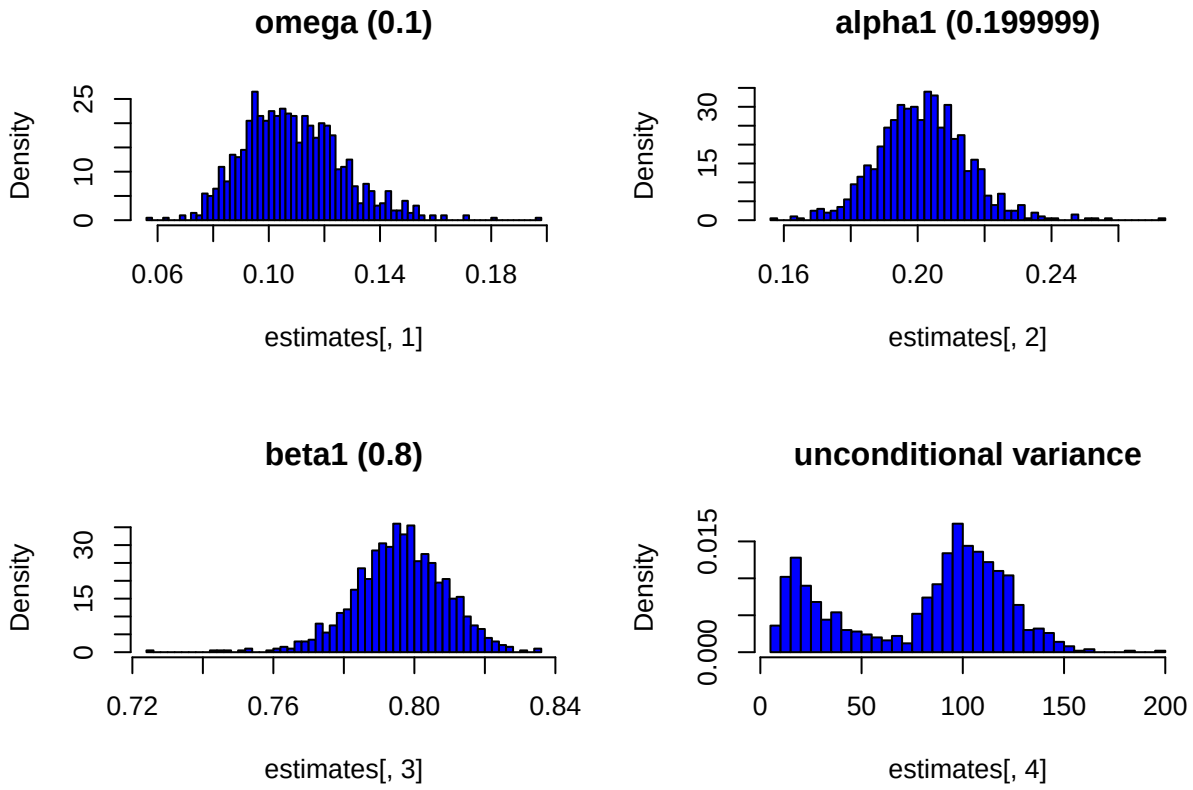
```r
#define parameters
n <- 5000
M <- 1000
mu <- 0
omega <- 0.1
alpha1 <- 0.199999
beta1 <- 0.8
#initialize matrix to save values
estimates <- matrix(ncol=4, nrow=M)
colnames(estimates) <- c("omega", "alpha1", "beta1", "uncon_var")
#specify GARCH(1,1) with predefined parameters
garch11_fixed_pars_spec <- ugarchspec(variance.model=list(garchOrder=c(1,1)),
                                      mean.model=list(armaOrder=c(0,0), include.mean=TRUE),
                                      fixed.pars=list(mu=mu, omega=omega, alpha1=alpha1, beta1=beta1),
                                      distribution.model="norm")
#specify GARCH(1,1) without predefined parameters
garch11_spec <- ugarchspec(variance.model=list(garchOrder=c(1,1)),
                           mean.model=list(armaOrder=c(0,0), include.mean=TRUE),
                           distribution.model="norm")
for(i in 1:M){
  #simulate 5000 observations of specified GARCH
  garch11_sim=ugarchpath(garch11_fixed_pars_spec, n.sim=n)
  #fit the simulated on values onto a GARCH(1,1)
  garch11_fit <- ugarchfit(garch11_spec, garch11_sim@path[["seriesSim"]])
```

```
  estimates[i, "omega"] <- garch11_fit@fit$robust.matcoef["omega", 1]
  estimates[i, "alpha1"] <- garch11_fit@fit$robust.matcoef["alpha1", 1]
  estimates[i, "beta1"] <- garch11_fit@fit$robust.matcoef["beta1", 1]
  estimates[i, "uncon_var"] <- estimates[i, "omega"]/(1-estimates[i, "alpha1"]-estimates[i, "beta1"])
}
```

5.2 Visualizing the parameters and the unconditional variance

The parameter estimates (ω, α_1, β_1) and the unconditional variance are visualized through histograms to understand their distribution across multiple simulation runs. These visualizations provide insights into the accuracy of parameter estimation and the stationarity condition under the assumed model.

```
par(mfrow=c(2,2))
hist(estimates[,1], breaks=50, freq=FALSE, main="omega (0.1)", col="blue")
hist(estimates[,2], breaks=50, freq=FALSE, main="alpha1 (0.199999)", col="blue")
hist(estimates[,3], breaks=50, freq=FALSE, main="beta1 (0.8)", col="blue")
hist(estimates[,4], breaks=50, freq=FALSE, main="unconditional variance", col="blue")
```



5.3 Interpreting the estimators

Using the true parameter values $\omega = 0.1$, $\alpha_1 = 0.199999$, and $\beta_1 = 0.8$, the real unconditional variance of the GARCH(1,1) process is calculated as:

$$\frac{0.1}{1 - 0.199999 - 0.8} = \frac{0.1}{0.000001} = 100,000. \tag{8}$$

The true unconditional variance is large due to the model's proximity to the stationarity boundary, where $1 - \alpha_1 - \beta_1$ approaches zero. The simulation, however, reveals a bimodal distribution of the estimated unconditional variance.

The first peak of the distribution, around 10, reflects cases where the estimated parameters deviate substantially from the true values, leading to a larger denominator and a smaller unconditional variance value. The second peak, near 110, corresponds to simulations where the estimated parameters are closer to their true values, producing a statistic closer to the theoretical value. The trough between 50 and 70 represents a transition zone in the parameter space that occurs less frequently.